**Week-1**

**Design Pattern and Principles**

**Exercise 1: Implementing the Singleton Pattern**

**Scenario:**

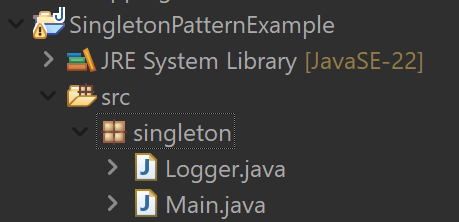
You need to ensure that a logging utility class in your application has only one instance throughout the application lifecycle to ensure consistent logging.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **SingletonPatternExample**.
2. **Define a Singleton Class:**
   * Create a class named Logger that has a private static instance of itself.
   * Ensure the constructor of Logger is private.
   * Provide a public static method to get the instance of the Logger class.
3. **Implement the Singleton Pattern:**
   * Write code to ensure that the Logger class follows the Singleton design pattern.
4. **Test the Singleton Implementation:**
   * Create a test class to verify that only one instance of Logger is created and used across the application.

**Implementation:**

**Folder Structure:**

****

**Logger.java**

// Logger.java

package singleton;

public class Logger {

private static Logger *instance*=new Logger();;

private Logger() {

System.*out*.println("Logger Initialized.");

}

public static Logger getInstance() {

return *instance*;

}

}

**Main.java**

//Main.java

package singleton;

public class Main {

public static void main(String[] args) {

// Fetching Logger instances

Logger logger1 = Logger.*getInstance*();

Logger logger2 = Logger.*getInstance*();

// Verifying Singleton behavior

if (logger1 == logger2) {

System.*out*.println("Same Logger instance used. Singleton works.");

} else {

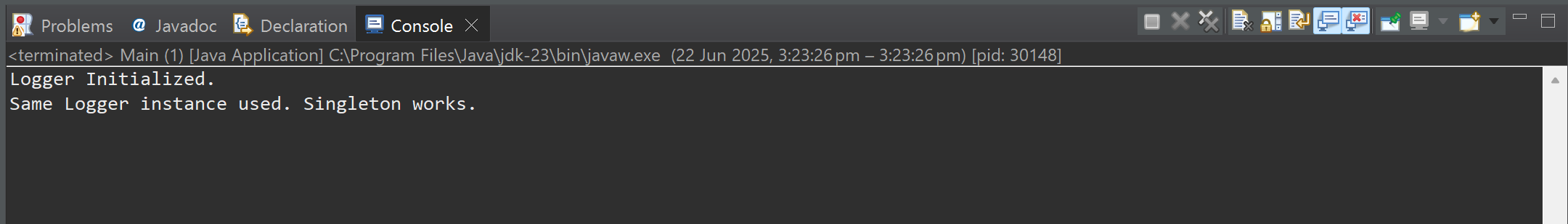
System.*out*.println("Different Logger instances. Singleton failed.");

}

}

}

**Output:**



**Exercise 2: Implementing the Factory Method Pattern**

**Scenario:**

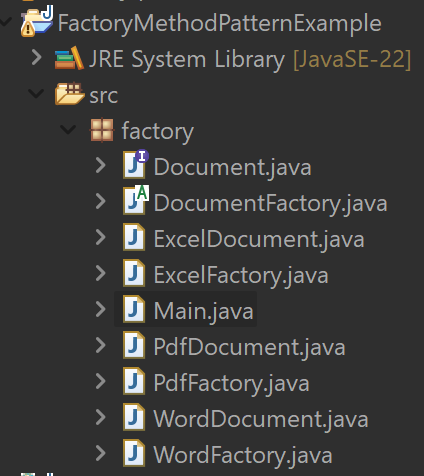
You are developing a document management system that needs to create different types of documents (e.g., Word, PDF, Excel). Use the Factory Method Pattern to achieve this.

**Steps:**

1. **Create a New Java Project:**
   * Create a new Java project named **FactoryMethodPatternExample**.
2. **Define Document Classes:**
   * Create interfaces or abstract classes for different document types such as **WordDocument**, **PdfDocument**, and **ExcelDocument**.
3. **Create Concrete Document Classes:**
   * Implement concrete classes for each document type that implements or extends the above interfaces or abstract classes.
4. **Implement the Factory Method:**
   * Create an abstract class **DocumentFactory** with a method **createDocument()**.
   * Create concrete factory classes for each document type that extends DocumentFactory and implements the **createDocument()** method.
5. **Test the Factory Method Implementation:**
   * Create a test class to demonstrate the creation of different document types using the factory method.

**Implementation:**

**Folder Structure:**

****

**Document.java**

//Document.java

package factory;

public interface Document {

void open();

}

**DocumentFactory.java**

//DocumentFactory.java

package factory;

public abstract class DocumentFactory {

public abstract Document createDocument();

}

**WordDocument.java**

//WordDocument.java

package factory;

public class WordDocument implements Document {

*@Override*

public void open() {

System.*out*.println("Opening a Word document.");

}

}

**WordFactory.java**

//WordFactory.java

package factory;

public class WordFactory extends DocumentFactory{

*@Override*

public Document createDocument() {

return new WordDocument();

}

}

**PdfDocument.java**

// PdfDocument.java

package factory;

public class PdfDocument implements Document{

*@Override*

public void open() {

System.***out***.println("Opening a PDF document.");

}

}

**PdfFactory.java**

// PdfFactory.java

package factory;

public class PdfFactory extends DocumentFactory {

*@Override*

public Document createDocument() {

return new PdfDocument();

}

}

**ExcelDocument.java**

// ExcelDocument.java

package factory;

public class ExcelDocument implements Document {

*@Override*

public void open() {

System.***out***.println("Opening an Excel document.");

}

}

**ExcelFactory.java**

// ExcelFactory.java

package factory;

public class ExcelFactory extends DocumentFactory{

*@Override*

public Document createDocument() {

return new ExcelDocument();

}

}

**Main.java**

// Main.java

package factory;

public class Main {

public static void main(String[] args) {

DocumentFactory wordFactory = new WordFactory();

Document wordDoc = wordFactory.createDocument();

wordDoc.open();

DocumentFactory pdfFactory = new PdfFactory();

Document pdfDoc = pdfFactory.createDocument();

pdfDoc.open();

DocumentFactory excelFactory = new ExcelFactory();

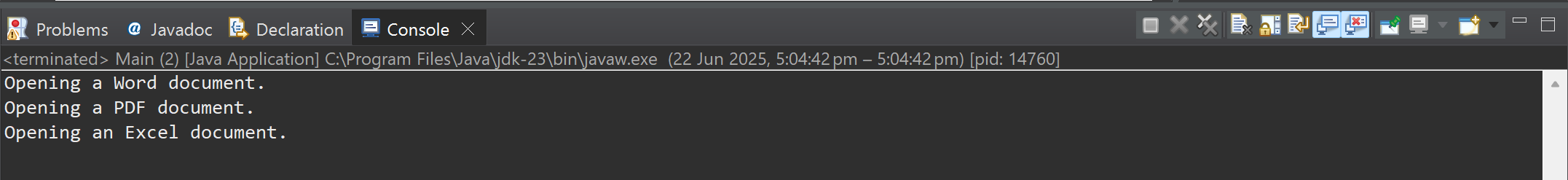
Document excelDoc = excelFactory.createDocument();

excelDoc.open();

}

}

**Output:**

****

**-- THE END --**